プログラミング入門1

第8回

メソッド（2）

授業開始前に自己点検

前回までの必須課題はすべてできていますか

前回までの学習項目であいまいな所はありませんか

理解できたかどうかは自分自身の基準をもとう

前回のテーマ

- メソッドとは
  - いくつかの命令の列を束ねて、一つの命令として扱えるようにしたもの
  - 今回学ぶメソッドの役割は、その他のプログラミング言語では関数またはサブルーチンと呼ばれることがある
- メソッドを書く
  - 宣言あるいは定義
- メソッドを使う
  - 起動あるいは呼び出し(call)

以下の質問に答えられますか？

- メソッドの宣言とは、起動とは何ですか
- メソッドの宣言はどのように書きますか
- メソッドの宣言はどこに置きますか
- メソッドの起動はどのようにしますか
- メソッドの仮引数、実引数とは何ですか
- メソッドの起動にあたって実引数はどのようにして仮引数に渡されますか
- 引数つきのメソッドの宣言はどのようにしますか
- 変数のスコープとは何ですか

メソッドの宣言とは、起動とは何ですか？

- メソッドの宣言とは、起動とは何ですか
- メソッドの宣言はどのように書きますか
- メソッドの宣言はどこに置きますか
- メソッドの起動はどのようにしますか
- メソッドの仮引数、実引数とは何ですか
- メソッドの起動にあたって実引数はどのようにして仮引数に渡されますか
- 引数つきのメソッドの宣言はどのようにしますか
- 変数のスコープとは何ですか

メソッドの宣言（定義）：一連の命令を束ねたものに名前をつけること

メソッドの起動（呼び出し）：メソッドを実行すること

注意：宣言しただけでは実行されない

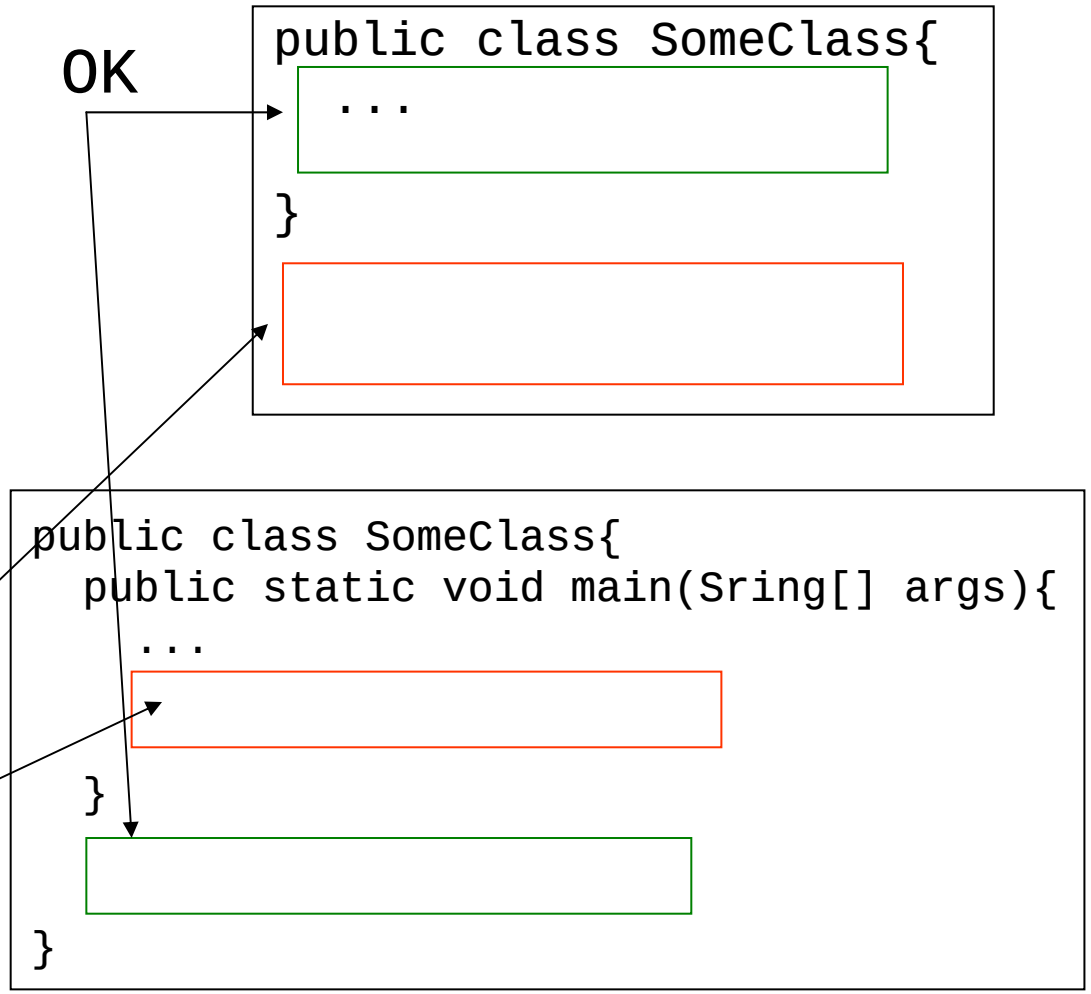
メソッドの宣言はどのように書きますか？

- メソッドの宣言とは、起動とは何ですか
- **メソッドの宣言はどのように書きますか**
- メソッドの宣言はどこに置きますか
- メソッドの起動はどのようにしますか
- メソッドの仮引数、実引数とは何ですか
- メソッドの起動にあたって実引数はどのようにして仮引数に渡されますか
- 引数つきのメソッドの宣言はどのようにしますか
- 変数のスコープとは何ですか

```
public static void <メソッドの名前> () {
  <メソッド本体>
  ここに命令文を並べる
}
```

メソッドの宣言はどこに置きますか？

- メソッドの宣言とは、起動とは何ですか
- メソッドの宣言はどのように書きますか
- **メソッドの宣言はどこに置きますか**
- メソッドの起動はどのようにしますか
- メソッドの仮引数、実引数とは何ですか
- メソッドの起動にあたって実引数はどのようにして仮引数に渡されますか
- 引数つきのメソッドの宣言はどのようにしますか
- 変数のスコープとは何ですか



メソッドの起動はどのようにしますか？

- メソッドの宣言とは、起動とは何ですか
- メソッドの宣言はどのように書きますか
- メソッドの宣言はどこに置きますか
- **メソッドの起動はどのようにしますか**
- メソッドの仮引数、実引数とは何ですか
- メソッドの起動にあたって実引数はどのようにして仮引数に渡されますか
- 引数つきのメソッドの宣言はどのようにしますか
- 変数のスコープとは何ですか

[引数なしの場合は]メソッドを実行したい場所で

メソッド名();

のように書く。

メソッドの仮引数、実引数とは何ですか？
メソッドの起動にあたって実引数はどのようにして仮引数に渡されますか？

- メソッドの宣言とは、起動とは何ですか
- メソッドの宣言はどのように書きますか
- メソッドの宣言はどこに置きますか
- メソッドの起動はどのようにしますか
- **メソッドの仮引数、実引数とは何ですか**
- **メソッドの起動にあたって実引数はどのようにして仮引数に渡されますか**
- 引数つきのメソッドの宣言はどのようにしますか
- 変数のスコープとは何ですか

```
public class Adder {
  public static void add(double x, double y) {
    System.out.println(x + y);
  }

  public static void main(String[] args) {
    add(10.5, 12.3);
    add(-2.3, 2.4);
  }
}
```

値渡し

引数つきのメソッドの宣言はどのようにしますか？

- メソッドの宣言とは、起動とは何ですか
- メソッドの宣言はどのように書きますか
- メソッドの宣言はどこに置きますか
- メソッドの起動はどのようにしますか
- メソッドの仮引数、実引数とは何ですか
- メソッドの起動にあたって実引数はどのようにして仮引数に渡されますか
- **引数つきのメソッドの宣言はどのようにしますか**
- 変数のスコープとは何ですか

```
public static void <メソッドの名前> (<仮引数の宣言>) {
  <メソッド本体>
}
```

あるいは「,」で区切られた複数の仮引数の宣言

変数のスコープとは何ですか

- メソッドの宣言とは、起動とは何ですか
- メソッドの宣言はどのように書きますか
- メソッドの宣言はどこに置きますか
- メソッドの起動はどのようにしますか
- メソッドの仮引数、実引数とは何ですか
- メソッドの起動にあたって実引数はどのようにして仮引数に渡されますか
- 引数つきのメソッドの宣言はどのようにしますか
- 変数のスコープとは何ですか

```
for (int i = 0; i < 10; i++) {
 // ...
}
// ERROR: ここで変数 i は使用できない
System.out.println(i);
```

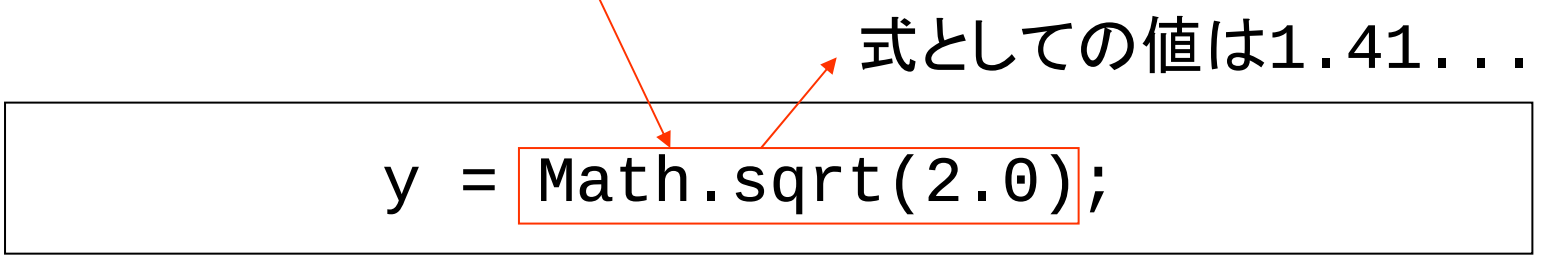
```
// i を for の外で宣言することによって、
// for の外側でも利用できるようにする
int i;
for (i = 0; i < 10; i++) {
 // ...
}
// これなら変数 i を使用できる
System.out.println(i);
```

もう答えられますね

- メソッドの宣言とは、起動とは何ですか
- メソッドの宣言はどのように書きますか
- メソッドの宣言はどこに置きますか
- メソッドの起動はどのようにしますか
- メソッドの仮引数、実引数とは何ですか
- メソッドの起動にあたって実引数はどのようにして仮引数に渡されますか
- 引数つきのメソッドの宣言はどのようにしますか
- 変数のスコープとは何ですか

今回のテーマ

- 式としてのメソッド
  - メソッドの起動では命令列が実行されるだけでなく、値をもつ（値を返す）ということ
  - 前回の講義では、メソッドを「いくつかの命令を束ねたもの」という視点で見ており、メソッドを起動することによって束ねられた命令の列を実行することができた。
  - 今回は、メソッドを「いくつかの命令を束ねた**式**」として扱う。



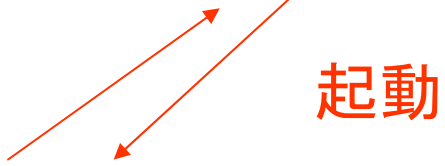
値をもつ、返すとは
（Javaに元々用意されている例から）

```
y = Math.sqrt(2.0);
```

1. Math (というclass) の sqrt というメソッドを使っている。
2. カッコ内がこのメソッドに渡される引数 (ひきすう:パラメタ parameter) であり、今の場合 double 型の実数 2.0 が引数である。
3. この文を実行すると 2.0 を引数としてメソッド sqrt が実行される (メソッドを起動する、呼び出すとも言う)。
4. その実行が終わると 1.4142135623730951 が返される（1.4142135623730951 という戻り値を持ってメソッドから帰ってくる）。
5. その値が変数 y に代入される。

Math.sqrtのような値を返すメソッドを自分で定義（宣言）する。ここでは平方を返すメソッドを考える。

```
public class SquareMethod {
  public static void main(String[] args) {
    for (int i = 1; i <= 5; i++) {
      System.out.println(square(i));
    }
  }

  public static int square(int x) {
    return x * x;
  }
}
```

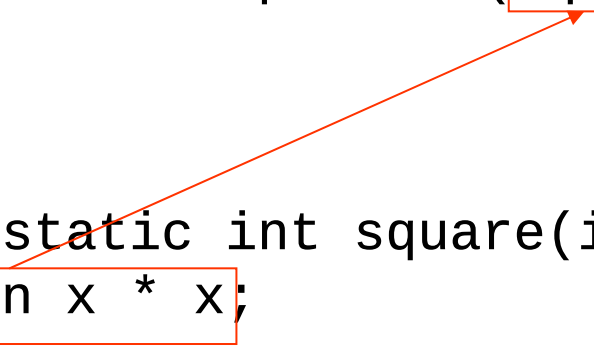


メソッドを呼び出すと、実引数の値が仮引数に代入されてメソッドの本体が実行される。

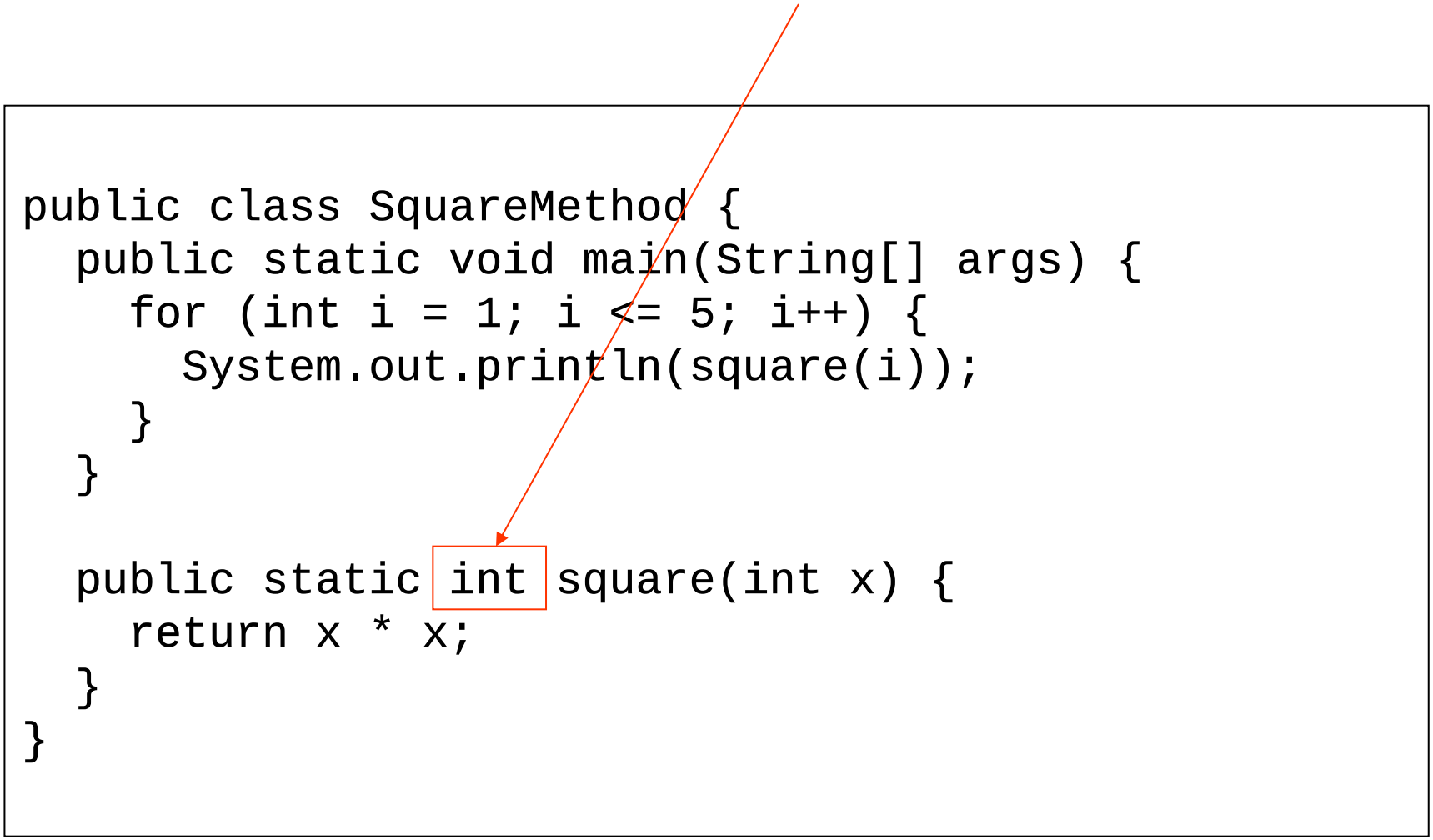
```
public class SquareMethod {
  public static void main(String[] args) {
    for (int i = 1; i <= 5; i++) {
      System.out.println(square(i));
    }
  }

  public static int square(int x) {
    return x * x;
  }
}
```

return文で返された戻り値がそのメソッド呼び出しの値となる。

```
public class SquareMethod {
  public static void main(String[] args) {
    for (int i = 1; i <= 5; i++) {
      System.out.println(square(i));
    }
  }

  public static int square(int x) {
    return x * x;
  }
}
```



戻り値の型



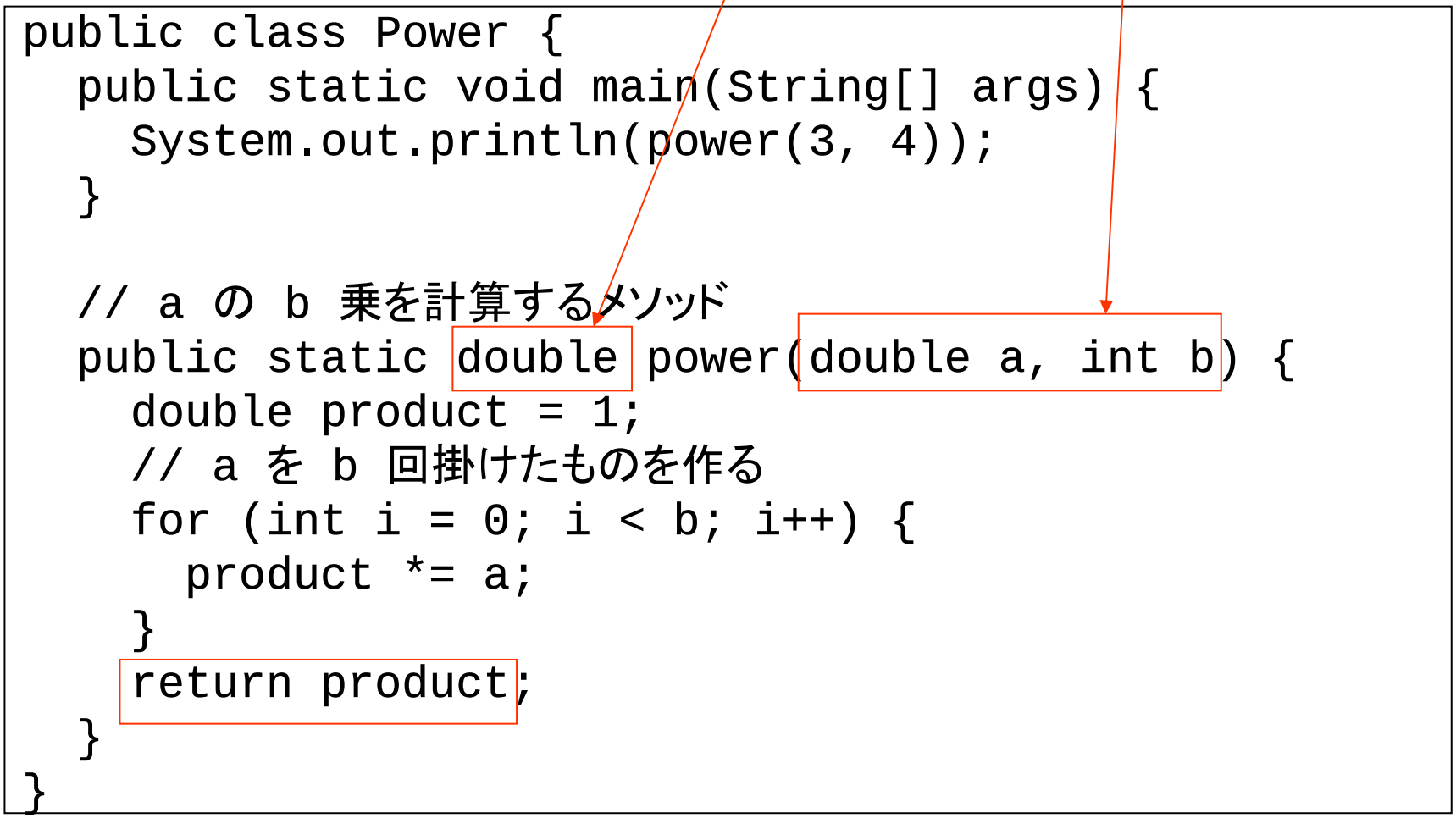
```
public class SquareMethod {
  public static void main(String[] args) {
    for (int i = 1; i <= 5; i++) {
      System.out.println(square(i));
    }
  }

  public static int square(int x) {
    return x * x;
  }
}
```

メソッド定義の書き方

```
public static 戻り値の型 メソッド名(仮引数の宣言の列){
  メソッドの本体
}
```

- 前回の講義では「戻り値の型」の部分が **void** であった。
- void とは「何もない」を表す型で、そう考えると戻り値型が void のメソッドを「何もないを返す (値を何も返さない)」とすることができる。
- 何か値を返すメソッドを作りたい場合、戻り値の型に int や double などを入れてやればよい。

実数と整数 (double, int)の2つの引数を取り、double 型の値を返すメソッド power



```
public class Power {
  public static void main(String[] args) {
    System.out.println(power(3, 4));
  }

  // a の b 乗を計算するメソッド
  public static double power(double a, int b) {
    double product = 1;
    // a を b 回掛けたものを作る
    for (int i = 0; i < b; i++) {
      product *= a;
    }
    return product;
  }
}
```

基本事項の確認: a^b を計算するのに迷わずこのようなループが書けますか

```
public class Power {
  public static void main(String[] args) {
    System.out.println(power(3, 4));
  }

  // a の b 乗を計算するメソッド
  public static double power(double a, int b) {
    double product = 1;
    // a を b 回掛けたものを作る
    for (int i = 0; i < b; i++) {
      product *= a;
    }
    return product;
  }
}
```

return 文は、戻り値の型が void 以外のときにメソッドを呼び出した元に値を返す命令

```
public static int square(int x) {
  return x * x;
}
```

戻り値がvoidではメソッドの途中で実行を打ち切りたいときに使える

```
public static void printOnNotZero(int x) {
  if (x == 0) {
    return;
  }
  System.out.println(x);
}
```

戻り値の型が void であるメソッドの呼び出しは、式の一部あるいは式が来るべき場所で使ってはいけない

コンパイルエラーになる

```
public class SomeClass {
  public static void main(String[] args) {
    System.out.println(voidMethod() + 5);
    System.out.println(voidMethod());
  }

  public static void voidMethod() {
    System.out.println(10);
  }
}
```

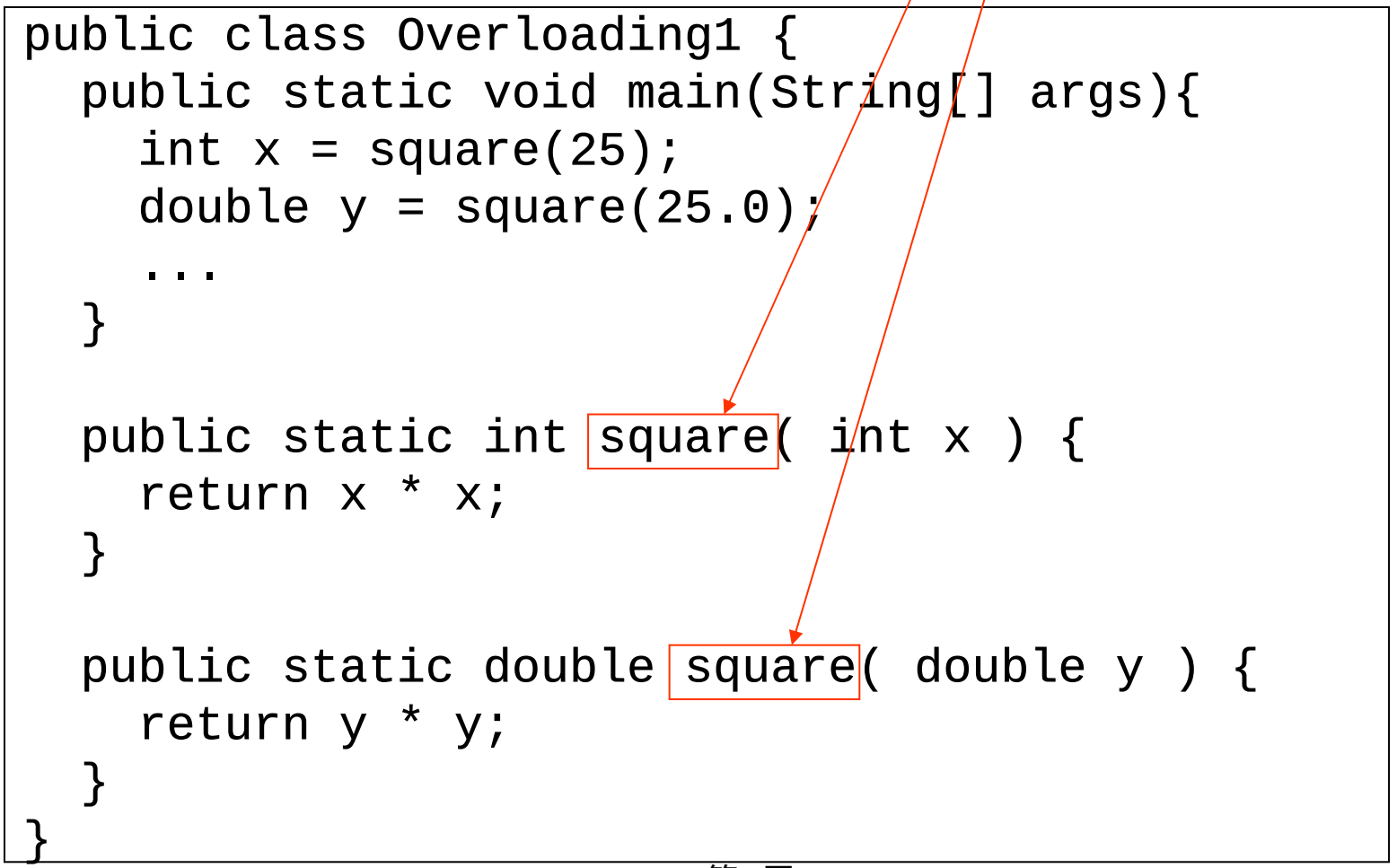
戻り値の型が void 以外であるメソッドの呼び出しは、通常は式の一部として使われるが、それだけで命令文としてもよい

戻り値を利用していないが、誤りではない。

```
public class SomeClass {
  public static void main(String[] args) {
    intMethod();
  }

  public static int intMethod() {
    System.out.println(10);
    return 0;
  }
}
```

メソッドの多重定義(overloading)

戻り値や仮引数の型が違えば、名前が同じでも異なるメソッドとなる



```
public class Overloading1 {
  public static void main(String[] args){
    int x = square(25);
    double y = square(25.0);
    ...
  }

  public static int square( int x ) {
    return x * x;
  }

  public static double square( double y ) {
    return y * y;
  }
}
```

```
public class Overloading2 {
  public static void main(String[] args) {
    // max(int)
    System.out.println(max(10));
    // max(int, int)
    System.out.println(max(10, 20));
    // max(int, int, int)
    System.out.println(max(10, 30, 20));
  }
  // 1つの中の最大
  public static int max(int a) {
    return a;
  }
  // 2つの中の最大
  public static int max(int a, int b) {
    if (a > b) {
      return a;
    } else {
      return b;
    }
  }
  // 3つの中の最大
  public static int max(int a, int b, int c) {
    // max(int, int) を 2回使う
    return max(a, max(b, c));
  }
}
```

引数の個数が違っても
異なるメソッド

演習に入る前に

- AddSubとCombinationという2つのプログラムを各自作成し実行、テストまでを一斉にやる
- 頃合をみて2つのプログラムの詳細な解説をするが
- 講義資料の解説を見ながらその意味を自分でよく考えること
- この2つのプログラムの意味をきちんと理解できた人だけが課題に進んでよい

一緒にやってみよう

- 今回の演習で使うテストドライバをいつものようにインストールする
  - ライブラリのアップデートがあるので手順を正確に実行すること
  - テストドライバの導入に成功すると
    - プロジェクト「java20XX」の中の「test」というフォルダに「08」という名前のフォルダが作成される。
    - このフォルダには今週使用するテスト一式が入っている。
- j1.lesson08 というパッケージを作成する
- 講義資料にあるAddSub, Combinationというプログラムを、このパッケージに作成する
  - 講義資料にある手順でテスト、実行までやること

AddSubの解説

```
public class AddSub{
  public static void main(String[] args) {
    System.out.println(add(1, 2));
    System.out.println(sub(10, 7));
    System.out.println(sub(add(1, 2), sub(3, 4)));
  }

  public static int add(int a, int b) {
    return a + b;
  }

  public static int sub(int a, int b) {
    return a - b;
  }
}
```

引数を1,2としてメソッド add(int a, int b) を呼び出し、戻り値を表示する

```
public class AddSub{
  public static void main(String[] args) {
    System.out.println(add(1, 2));
    System.out.println(sub(10, 7));
    System.out.println(sub(add(1, 2), sub(3, 4)));
  }

  public static int add(int a, int b) {
    return a + b;
  }

  public static int sub(int a, int b) {
    return a - b;
  }
}
```

引数を10,7としてメソッド sub(int a, int b) を呼び出し、戻り値を表示する

```
public class AddSub{
  public static void main(String[] args) {
    System.out.println(add(1, 2));
    System.out.println(sub(10, 7));
    System.out.println(sub(add(1, 2), sub(3, 4)));
  }

  public static int add(int a, int b) {
    return a + b;
  }

  public static int sub(int a, int b) {
    return a - b;
  }
}
```

引数を1,2としてメソッド add(int a, int b) を呼び出し

```
public class AddSub{
  public static void main(String[] args) {
    System.out.println(add(1, 2));
    System.out.println(sub(10, 7));
    System.out.println(sub(add(1, 2), sub(3, 4)));
  }

  public static int add(int a, int b) {
    return a + b;
  }

  public static int sub(int a, int b) {
    return a - b;
  }
}
```

さらに引数を3,4としてメソッド sub(int a, int b) を呼び出す

```
public class AddSub{
  public static void main(String[] args) {
    System.out.println(add(1, 2));
    System.out.println(sub(10, 7));
    System.out.println(sub(add(1, 2), sub(3, 4)));
  }

  public static int add(int a, int b) {
    return a + b;
  }

  public static int sub(int a, int b) {
    return a - b;
  }
}
```

それらの戻り値を引数としてメソッド sub(int a, int b) を呼び出し

```
public class AddSub{
  public static void main(String[] args) {
    System.out.println(add(1, 2));
    System.out.println(sub(10, 7));
    System.out.println(sub(add(1, 2), sub(3, 4)));
  }

  public static int add(int a, int b) {
    return a + b;
  }

  public static int sub(int a, int b) {
    return a - b;
  }
}
```

計算結果を表示する

```
public class AddSub{
  public static void main(String[] args) {
    System.out.println(add(1, 2));
    System.out.println(sub(10, 7));
    System.out.println(sub(add(1, 2), sub(3, 4)));
  }

  public static int add(int a, int b) {
    return a + b;
  }

  public static int sub(int a, int b) {
    return a - b;
  }
}
```

メソッドの宣言の書き方

```
public class AddSub{
  public static void main(String[] args) {
    System.out.println(add(1, 2));
    System.out.println(sub(10, 7));
    System.out.println(sub(add(1, 2), sub(3, 4)));
  }

  public static int add(int a, int b) {
    return a + b;
  }

  public static int sub(int a, int b) {
    return a - b;
  }
}
```

Combinationの解説

```
public class Combination {
  public static void main(String[] args) {
    //combinationメソッドを呼び出す
    System.out.println("10人から2人選ぶ組み合わせは"
      + combination(10, 2) + "通り");
  }
  // nCr = n! / ((n-r)! * r!)を計算するメソッド
  public static int combination(int n, int r) {
    // 階乗の計算でfactメソッドを呼び出す
    return fact(n) / (fact(n-r) * fact(r));
  }
  //nの階乗を計算するメソッド
  public static int fact(int n) {
    int total = 1;
    for (int i = n; i >= 1; i--) {
      total *= i;
    }
    return total;
  }
}
```

階乗の計算は基本中の基本：n(n-1)(n-2)…1

```
public class Combination {
  public static void main(String[] args) {
    //combinationメソッドを呼び出す
    System.out.println("10人から2人選ぶ組み合わせは"
      + combination(10, 2) + "通り");
  }
  // nCr = n! / ((n-r)! * r!)を計算するメソッド
  public static int combination(int n, int r) {
    // 階乗の計算でfactメソッドを呼び出す
    return fact(n) / (fact(n-r) * fact(r));
  }
  //nの階乗を計算するメソッド
  public static int fact(int n) {
    int total = 1;
    for (int i = n; i >= 1; i--) {
      total *= i;
    }
    return total;
  }
}
```

階乗を使って:組合せの計算

```
public class Combination {
  public static void main(String[] args) {
    //combinationメソッドを呼び出す
    System.out.println("10人から2人選ぶ組み合わせは"
      + combination(10, 2) + "通り");
  }
  // nCr = n! / ((n-r)! * r!)を計算するメソッド
  public static int combination(int n, int r) {
    // 階乗の計算でfactメソッドを呼び出す
    return fact(n) / (fact(n-r) * fact(r));
  }
  //nの階乗を計算するメソッド
  public static int fact(int n) {
    int total = 1;
    for (int i = n; i >= 1; i--) {
      total *= i;
    }
    return total;
  }
}
```

引数を10,2としてメソッド combination(int n, int r) を呼び出し

```
public class Combination {
  public static void main(String[] args) {
    //combinationメソッドを呼び出す
    System.out.println("10人から2人選ぶ組み合わせは"
      + combination(10, 2) + "通り");
  }
  // nCr = n! / ((n-r)! * r!)を計算するメソッド
  public static int combination(int n, int r) {
    // 階乗の計算でfactメソッドを呼び出す
    return fact(n) / (fact(n-r) * fact(r));
  }
  //nの階乗を計算するメソッド
  public static int fact(int n) {
    int total = 1;
    for (int i = n; i >= 1; i--) {
      total *= i;
    }
    return total;
  }
}
```

出力:10人から2人選ぶ組み合わせは45通り

```
public class Combination {
  public static void main(String[] args) {
    //combinationメソッドを呼び出す
    System.out.println("10人から2人選ぶ組み合わせは"
      + combination(10, 2) + "通り");
  }
  // nCr = n! / ((n-r)! * r!)を計算するメソッド
  public static int combination(int n, int r) {
    // 階乗の計算でfactメソッドを呼び出す
    return fact(n) / (fact(n-r) * fact(r));
  }
  //nの階乗を計算するメソッド
  public static int fact(int n) {
    int total = 1;
    for (int i = n; i >= 1; i--) {
      total *= i;
    }
    return total;
  }
}
```

課題

各自のペースで
「第08週目の課題」をやってみよう

```
==== 課題0801
main の仕事
   (1) プロンプトを適切に出しながら、入力する
   (2) cels2fahr を適切な引数を与えて起動し、戻り値を変数にしまう
   (3) 出力する
          注:  (2),(3)を1つの式にしてもよい。

cels2fahr の仕事
   (1) 引数として受け取ったデータから華氏の値を計算し return する


==== 課題0802
main の仕事
   (1) プロンプトを適切に出しながら、入力する
   (2) distance を適切な引数を与えて起動し、戻り値を変数にしまう
   (3) 出力する
          注:  (2),(3)を1つの式にしてもよい。

distance の仕事
   (1) 引数として受け取ったデータから距離を計算し return する
```